**Stress-Map Documentation**

Pre-process extract

Pre-process the extract and start a routing engine HTTP server on port 5000.

Depending on what kind of routing you want to use, choose the profile accordingly:

car: car.lua

walking: foot.lua

bike: bicycle.lua

***docker run -t -v c:/docker:/data osrm/osrm-backend osrm-extract -p /opt/PROFILE.lua /data/CITY-latest.osm.pbf***

So for car rides in Berlin use:***docker run -t -v c:/docker:/data osrm/osrm-backend osrm-extract -p /opt/car.lua /data/berlin-latest.osm.pbf***

Then run:***docker run -t -v c:/docker:/data osrm/osrm-backend osrm-partition /data/berlin-latest.osrm***

***docker run -t -v c:/docker:/data osrm/osrm-backend osrm-customize /data/berlin-latest.osrm***

**Why we setup Docker Image**

Because extracting, customizing osm.pbf that is map of any particular region is not suitable for desktop machine requires lot of computational power and RAM so Docker

**How to Setup OSRM Docker Image**

1. Refer this article

<https://gist.github.com/AlexandraKapp/e0eee2beacc93e765113aff43ec77789>

* 1. Start by installing docker on your desktop
  2. Pulling the image
  3. Download OpenStreetMap extract (:I downaloaeded onlot Atlanta, I suggest to downalod all of United States of America - <https://download.geofabrik.de/north-america/us.html>
  4. Start Pre-Processing since it is important for to create paths for Bicycles, Foot and Car
  5. Start Routing Engine  
       
     docker run --name osrm -t -i -p 5000:5000 -v c:/docker:/data osrm/osrm-backend osrm-routed --algorithm mld /data/berlin-latest.osrm

* 1. To test request a based on profile use processed change driving to desired

curl "<http://127.0.0.1:5000/route/v1/driving/13.388860,52.517037;13.385983,52.496891?steps=true>"

* 1. Refer Installation of OSRM Server document pdf if faced with any issues
  2. Docker start “nameofyourdocker” Docker stop “nameofyourdocker”

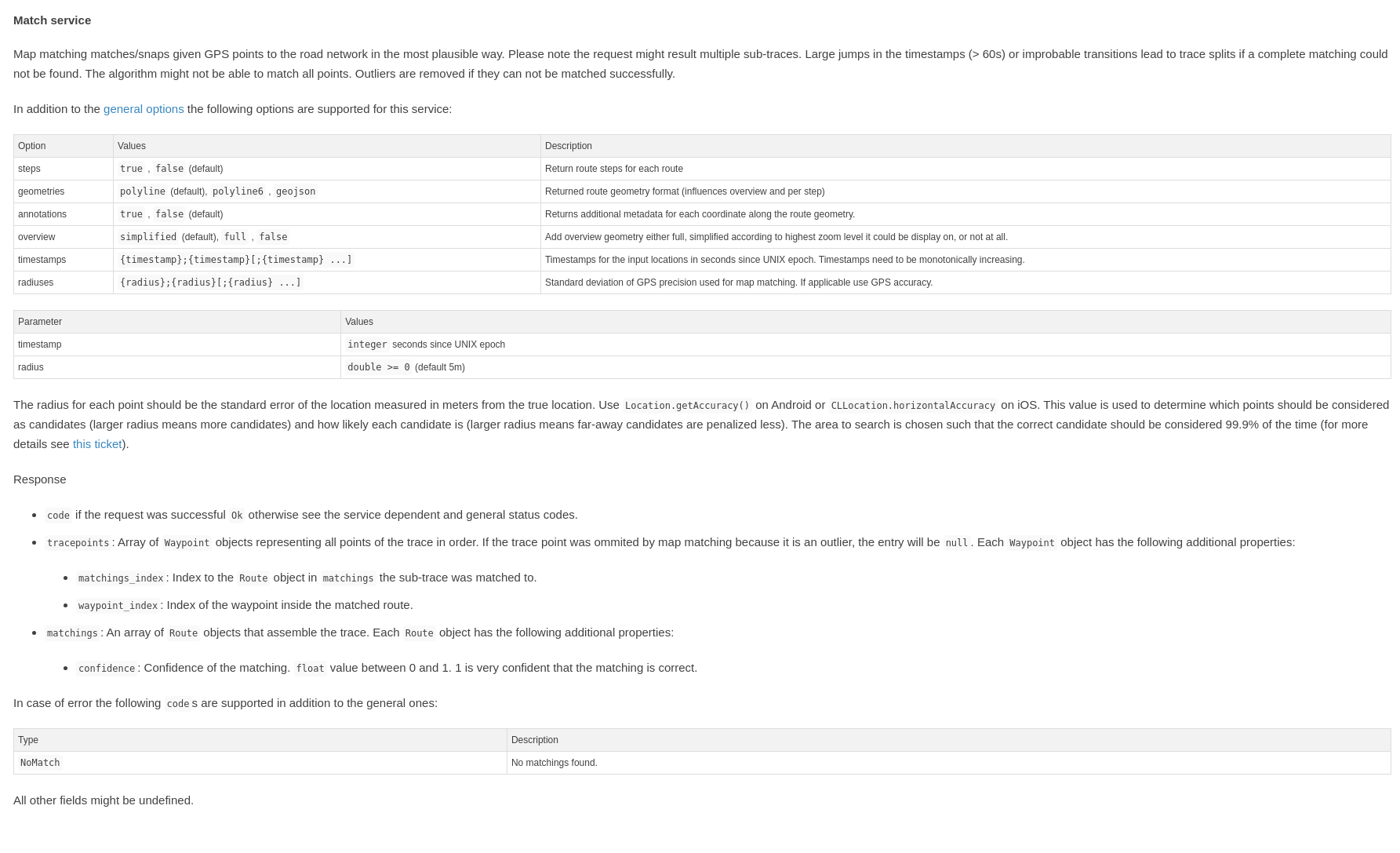
1. Other alternatives Valhalla / Mapbox / Graphhopper
   1. Documentation

**OSRM Open Source Routing Machine API**

OSRM Response

****

1. OSRM documentation can be found on this link - [https://project-osrm.org/docs/v5.5.1/api/#general-options](https://project-osrm.org/docs/v5.5.1/api/" \l "general-options)
   1. **Match Service** - [https://project-osrm.org/docs/v5.5.1/api/#match-service](https://project-osrm.org/docs/v5.5.1/api/" \l "match-service)



We used match service to align the group of coordiantes from SQL server to route it (snapped) most efficient way using this service. It take maximum of 90 to 100 coordiantes in one API call. Example

<http://localhost:5000/match/v1/bicycle/-84.365213,33.777787;-84.364595,33.777679?overview=full&radiuses=49;49&geometries=geojson&tidy=true&annotations=true>

Annotations: it consists metadata about the coordinates revived from API that is data about the snapped coordiantes to the road. It inlcudes nodes,

"nodes": [2256317495, 69331811, 8155111912],

"datasources": [0, 0],

"speed": [4.2, 4.2],

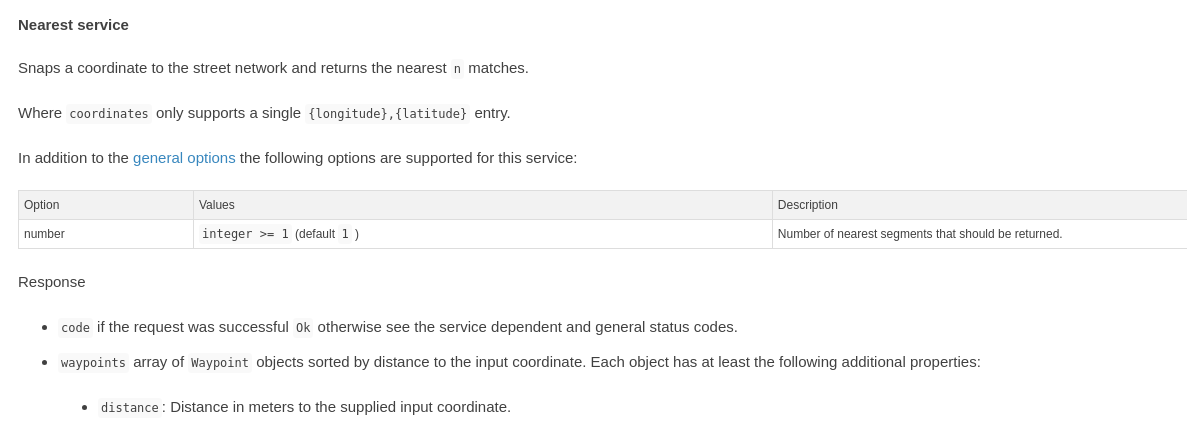
"weight": [14.5, 7.4],

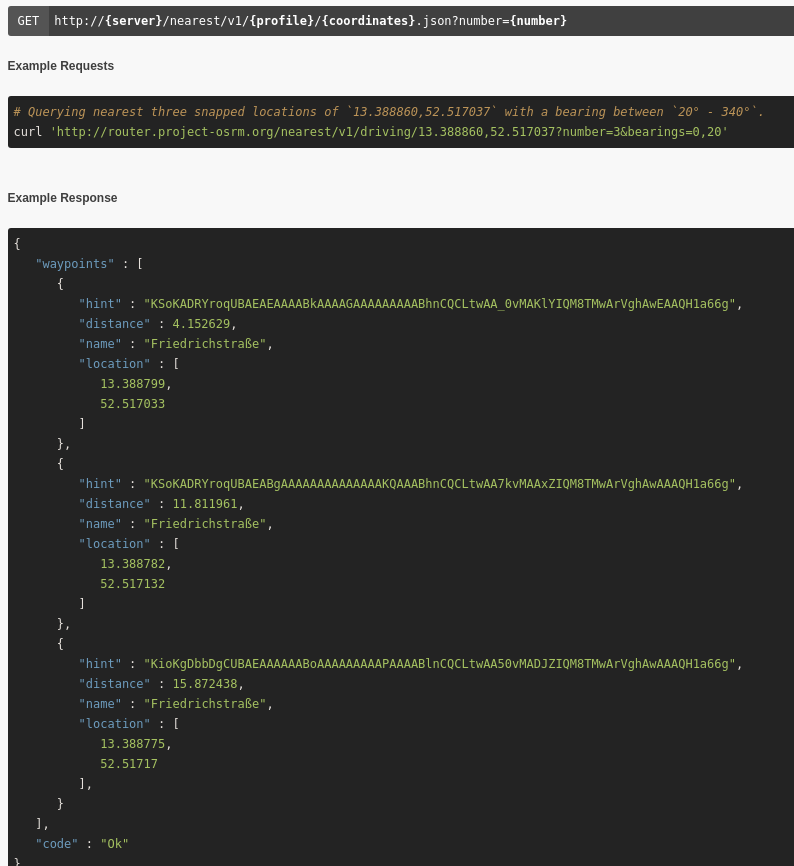
"duration": [14.5, 7.4],

"distance": [60.380708, 31.030934]

Properties

* distance: The distance, in metres, between each pair of coordinates
* duration: The duration between each pair of coordinates, in seconds
* datasources: The index of the datasource for the speed between each pair of coordinates. 0 is the default profile, other values are supplied via --segment-speed-file to osrm-contract
* nodes: The OSM node ID for each coordinate along the route, excluding the first/last user-supplied coordinates
  1. Nearest Service – It could be used to find out nearest ways, roads, paths





* 1. Route Service
  2. Trip Service
  3. Tile Service
  4. Table Service

**OSM Open Street Map API**

1. Open Street -

Communities to Join - <https://www.openstreetmap.org/communities>

**Architecture**

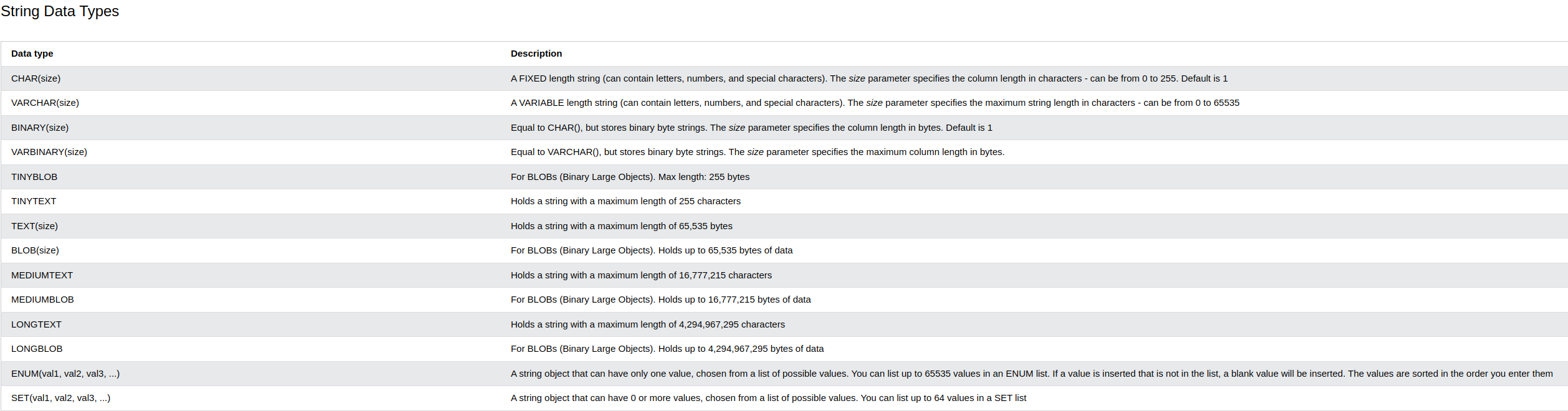
1. Polyglot
   1. Python & Py Notebooks
   2. Express, Node Javascript / Typescript
   3. SQL

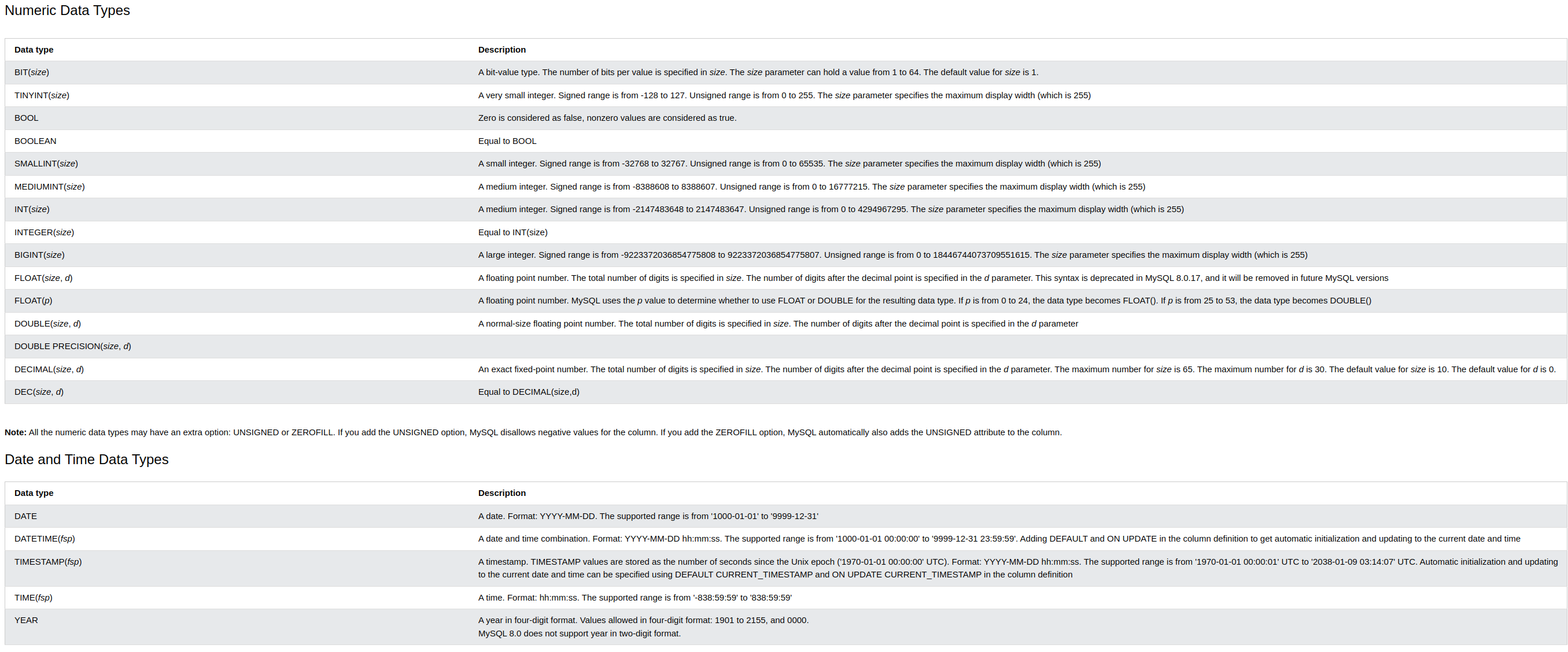
|  |  |  |  |
| --- | --- | --- | --- |
| Data Extraction and Analysis | Python & Py Notebooks (pyosrmplay2.pynb) | OSRM API for routing | OSM API for Nodes |
| MySQL Table (st – database, osr table) | | |
| Translation // Backend | Node & Express Javascript Loading MySQL table (server.js),  Previous works routing using API (server.js) | | |
| Frontend | Front end display of routes (MyMap.tsx, Snap.tsx) | | |

**SQL Table**

I am writing about datatypes here that we used to store the information from OSRM & OSM TEXT, LONGTEXT, VARCHAR, INT.

Below screenshots are about the description and size of data types for future reference.





**To start MySQL on Linux:**

**My System:** sudo mysql -u root -p (password is root)

**if not root just** press enter

USE DatabaseName;

Here, the **DatabaseName** is the name of the database that we want to select

CREATE DATABASE testDB;

SHOW DATABASES;

**to input database:** “soruce /filepath”

**To install & start XAMPP:**

How do I install XAMPP?

Choose your flavor for your linux OS, the 32-bit or 64-bit version.

Change the permissions to the installer

chmod 755 xampp-linux-\*-installer.run

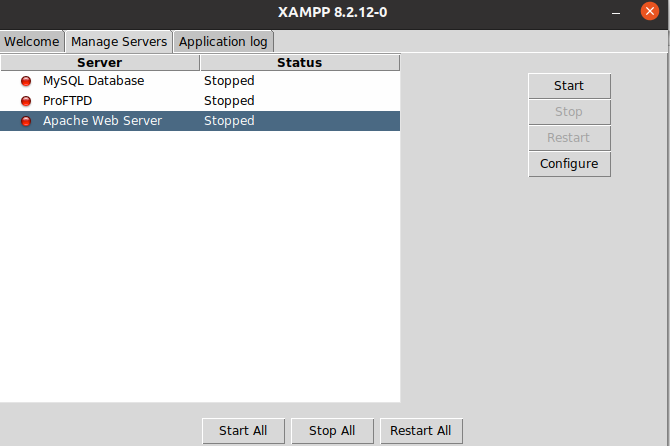
Run the installer

sudo ./xampp-linux-\*-installer.run

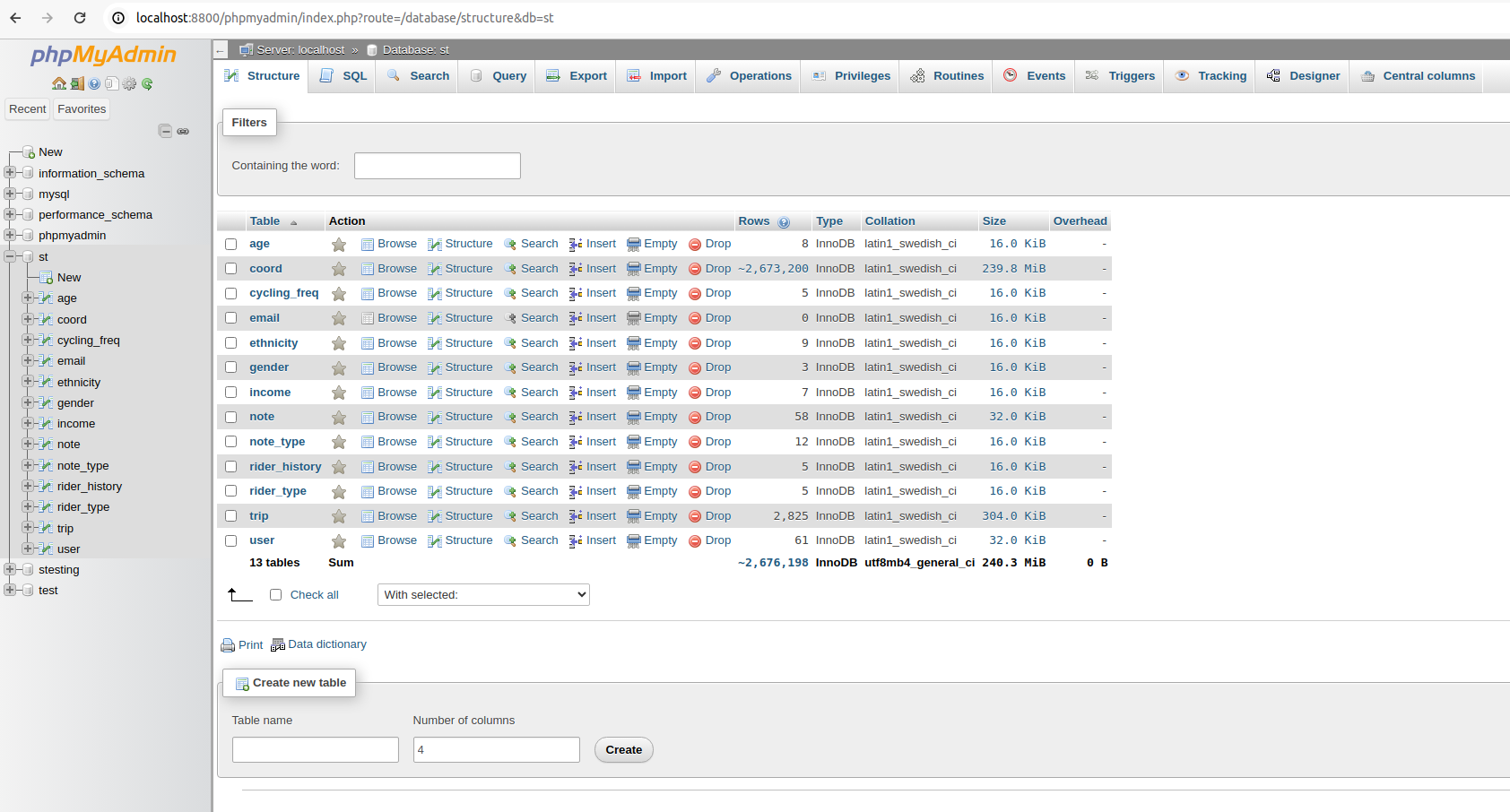
That's all. XAMPP is now installed below the /opt/lampp directory.

cd *opt*lampp/

sudo ./manager-linux-x64.run



start Apache Web Server and start MySQL Database then go to localhsot phpyadmin to access and see the SQL table



**To access XAMPP/MariaDB through terminal on Linux:**

“sudo /opt/lampp/bin/mysql -h localhost -u root -p”

**to input database:** “soruce /filepath”

**CODE**

Snap.tsx

MyMap.tsx

**Linux CMD Commands**

<https://www.linuxtrainingacademy.com/linux-commands-cheat-sheet/>

<https://www.reddit.com/r/linux/comments/12dll1h/linux_command_line_cheat_sheet_all_the_commands/>

<https://www.geeksforgeeks.org/linux-commands-cheat-sheet/>

history | grep mysql

history | grep docker

**To start MySQL:**

sudo mysql -u root -p (password is root)

**To start XAMPP:**

cd *opt*lampp/

sudo ./manager-linux-x64.run

**To access XAMPP/MariaDB through CMD:**

sudo /opt/lampp/bin/mysql -h localhost -u root -p

to input database: soruce …….filepath

**Download and Install XAMPP -** <https://www.youtube.com/watch?v=HJl2ILUfBoA>

<https://linux.how2shout.com/how-to-start-xampp-in-ubuntu-using-the-command-line/>

**Download Docker CE and Docker Desktop -** <https://www.youtube.com/watch?v=ILdziITdSag>

<https://www.youtube.com/watch?v=JsXNBIsFzu4>

**Few Common Erros Faced:**

Require Error:

*const express = require("express"); ^ ReferenceError: require is not defined in ES module scope, you can use import instead This file is being treated as an ES module because it has a '.js' file extension and '/backend/package.json' contains "type": "module". To treat it as a CommonJS script, rename it to use the '.cjs' file extension.*

<https://github.com/vercel/next.js/issues/24334>

<https://learn.coderslang.com/0021-nodejs-require-is-not-defined-error/>

**Open GraphHopper on Localhost:**

docker run -p 8989:8989 -v "$(pwd)/data:/data" israelhikingmap/graphhopper --input /data/georgia-latest.osm.pbf --host 0.0.0.0 --config /data/config-graphhopper.yml